Dialogue Efficiency Evaluation of Turn-Taking Phenomena in a Multi-layer Incremental Simulated Environment

Chapter · July 2015

DOI: 10.1007/978-3-319-21380-4_127

3 authors, including:

Hatim Khouzaimi

Orange Labs / Laboratoire Informatique d'Avig…

12 PUBLICATIONS **42** CITATIONS

SEE PROFILE

Romain Laroche

Microsoft Maluuba

58 PUBLICATIONS **185** CITATIONS

SEE PROFILE

Dialogue Efficiency Evaluation of Turn-Taking Phenomena in a Multi-Layer Incremental Simulated Environment

Hatim Khouzaimi[1,2], Romain Laroche[1], and Fabrice Lefèvre[2]

[1] Orange Labs, Issy-les-Moulineaux, France,
[2] CERI-LIA, Univ. Avignon, France,
hatim.khouzaimi@orange.com, romain.laroche@orange.com,
fabrice.lefevre@univ-avignon.fr

Abstract. One of the main challenges when it comes to designing a dialogue system is error handling. The Automatic Speech Recognition (ASR) technology is not perfect and the user may use words or expressions that are unknown to the system (off-domain). Traditional dialogue systems wait until the end of the user's utterance before processing and cannot be interrupted after taking the floor. On the contrary, incremental dialogue systems process the user's speech signal on the fly. Therefore they are able to react quickly whenever an error is suspected. In this paper, we devise a simulated environment to show that incremental dialogue systems make it possible to design new strategies for efficient error handling.

Keywords: Spoken Dialogue Systems, Incremental Dialogue, User Simulator

1 Introduction

In the last few years, automatic speech recognition (ASR) technology has made a lot of progress so that it is no longer a bottleneck in the development of spoken dialogue systems. Some industrial services already have a vocal interaction mode but their efficiency and their naturalness are still poor. An important aspect that still needs to be improved is floor management. As for now, time is shared in a rigid way between players as the system has to wait for the user to finish its sentence before speaking and generally the opposite is true (though some barge-in capabilities are already available in some systems). This is not natural, and as we show in this paper, it is not the most efficient way of interaction. When human beings talk to each other, the listener understands the speaker on the fly and sometimes can even guess the remaining part of the utterance before it ends [10]. This ability is manifested through different turn-taking phenomena and some of the most commonly studied in the field of dialogue systems [9, 3, 8, 5] have been implemented in this work.

Systems that replicate this behaviour are called incremental dialogue systems. Some sequential architectures have been published in order to design such

systems [4, 1, 7]. In this paper we use a multi-layer architecture [6] where the turn-taking management part is separated from the traditional dialogue manager (DM). A *Scheduler* module is added as an extra-layer prior to the DM.

We show that incremental dialogue processing offers possibilities to make dialogue systems more robust to noise. To demonstrate our work, a user simulator has been developed in the domain of personal agenda assistant. Overall dialogue efficiency is measured through dialogue duration and task completion [11, 2]. Moreover, the turn-taking phenomena are evaluated separately in order to highlight the most important ones in terms of efficiency improvement.

Section 2 introduces the simulated environment and the chosen task. Section 3 describes the experiment and the results and finally, Section 4 concludes and announces future work.

2 Simulated environment

Three different modules compose the simulated environment: the user simulator, the Scheduler and the service. The service is a personal agenda manager. In the case of non-incremental strategies, the user simulator interacts with the service directly using natural language commands to accomplish three types of task: adding, modifying or deleting an event from the agenda. Therefore, a task is defined by four slots: the type of action (ADD, MODIFY or DELETE), the title of the event, its date and its time slot. Moreover, juxtapositions and overlaps between events are not tolerated and thus justify some interactions in case of conflicting commands. In order to simulate incremental strategies, a Scheduler has been added between the user simulator and the service as described in [6]. It is in charge of floor-taking decisions whereas the service handles more high-level semantic decisions (computing responses based on the user simulator's utterances).

Initially, the agenda contains few pre-defined events. Then the user simulator is given a list of events that should be added during the dialogue. Each event has a fixed priority and a set of alternative dates and time slots that can be used in case of conflict. The user simulator strategy is to to make the maximum number of events with the highest probability fit in the agenda.

2.1 User simulator

A simple algorithm has been implemented so that the user simulator can calculate the next action to perform. When trying to add a new event, if the slot is free then it is added, otherwise it checks the alternative slots. If they are all busy, it checks whether it is possible to move a conflicting event. If it is not possible, then the priority of the event to add is compared to the less important priority among the conflicting events. If it is greater, then the conflicting event is deleted and replaced with the new event, otherwise the latter is forgotten.

After the next user act is determined, it is incrementally sent to an ASR output simulator. The increment chosen here is the word, for the sake of simplicity

and as there is no need for greater precision in this work. Each word-based step is called a *micro-turn*. The latter maintains an N-Best list (the top recognition hypotheses with their confidence scores) corresponding to the partial utterance made so far by the user. A Word Error Rate (WER) parameter is set to control the noise level.

Real ASR modules are not cumulative. A new chunk of audio signal can modify an important part if not the whole ASR best output hypothesis. This is due to the fact that the language model suddenly recognises a pattern that is more likely to correspond to what the user just said. To simulate this phenomenon, whenever a new concept appears in an hypothesis, the corresponding confidence score is boosted (so it has a chance to be the top hypothesis). As a consequence, the last words of a partial request are more likely to change than the ones that appeared earlier in the utterance and no decision should be made based on them. Thus, the Scheduler removes a few last words from the current partial utterance and makes a decision based on the remaining prefix. The latter will be called *last stable partial utterance*. The number of words removed will be referred to as the *stability margin* (SM).

2.2 Turn-taking rules

Five turn-taking phenomena (requiring incremental processing) have been implemented in this work. Four of them require the system to make a decision (implemented as a set of rules added to the Scheduler) and one of them depend on the user's behaviour only. At each micro-turn, the Scheduler has to decide either to remain mute (WAIT), or to repeat the last word of the last stable utterance (REPEAT) or to retrieve the current response obtained from the service (SPEAK).

FAIL_RAW: While speaking, the user has no guarantee that the system understands her message (due to noise or use of off-domain words). Therefore, in order to prevent the user from speaking too long without being understood, if no key concept has been detected after a certain period of time, the Scheduler performs a SPEAK action (as no key concept is detected in the current partial utterance, the last service's response will be something like *Sorry. I don't understand.*). The threshold is set to 6 micro-turns for open questions and time slot questions, to 3 for yes/no questions and 4 for dates (some concepts need more words to be expressed) so that the user can utter some outof domain words and the key concepts have time to reach som stability.

INCOHERENCE_INTERP: Although understood, the user's utterance can still be problematic if it is incoherent with the dialogue context (trying to modify a non existing event in the agenda for example). Therefore, as soon as the service makes an incoherence alert, the Scheduler waits to get \$SM more words and if the partial request at time $t - \$SM$ is a prefix of the one at time t (no changes due to ASR instability) then a SPEAK is activated.

FEEDBACK_RAW: The last word's confidence score is estimated as the ratio between the last partial request score and the one before last. If this ratio is

below a given threshold, the last word is repeated (REPEAT action) $SM words later if it is still present in the partial utterance.

BARGE_IN_RESP (System): When there is enough information in the partial request for the service to generate a response, the system can barge-in before the user ends her utterance. The Scheduler performs a SPEAK $SM words later if the partial utterance that generates the response is a prefix of the current one (no changes due to ASR instability).

BARGE_IN_RESP (User): Unlike the others this phenomenon corresponds to a user's decision. In this work, we suppose that the user is familiar with the system to barge-in as soon as it has enough information without letting the system finish its utterance.

3 Experiments and Results

Several strategies have been implemented in the User simulator and in the Scheduler to measure the impact on the dialogue performance of the various floor-taking phenomena described in the previous section. With the system initiative strategy (SysIni), the user is asked for the different chunks of information one by one (action, description, date and time slot). On the contrary, with the user initiative one (UsrIni), all the necessary information must be given in one request. The mixed initiative strategies (MixIni) behaves like UsrIni and if it fails, it switches to SysIni. Finally, we introduced incremental behaviours (the five turn-taking phenomena described in 2.2) to both UsrIni (UsrIni+Incr) and MixIni (MixIni+Incr).

We define a scenario as two lists of events. The first one corresponds to the events already in the agenda before the dialogue whereas the second contains the events to add during the dialogue. Our experiments are based on three handcrafted scenarios. In order to analyse the effect of noise on these strategies, we vary the simulated WER between 0 and 0.3 with a step of 0.03. For each scenario and each WER, 1000 dialogues are simulated.

Fig. 1. Mean dialogue duration and task completion for generic strategies.

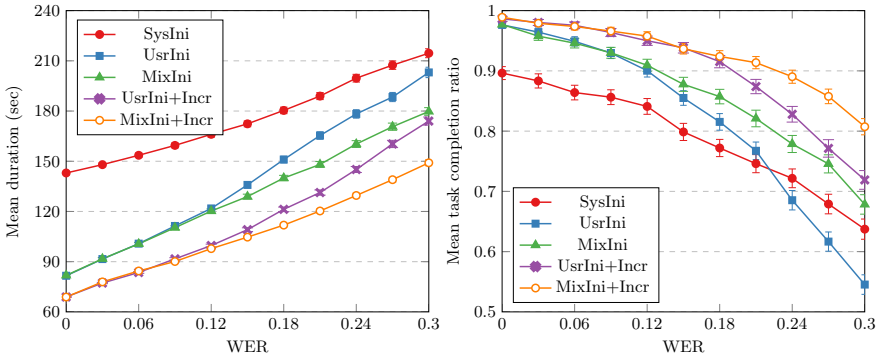
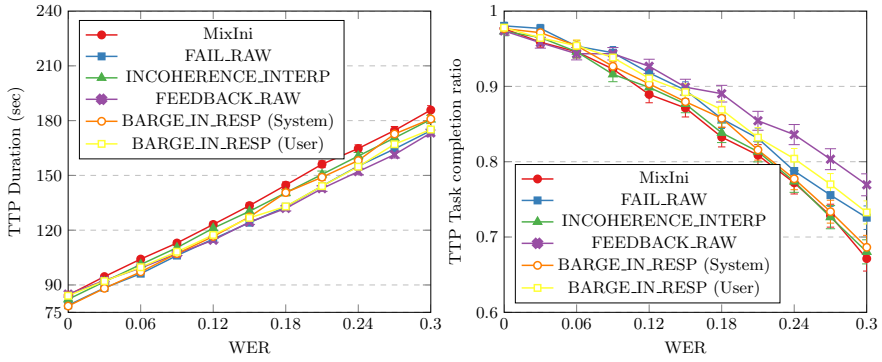


Fig. 2. Mean dialogue duration and task completion for different turn-taking phenomena.

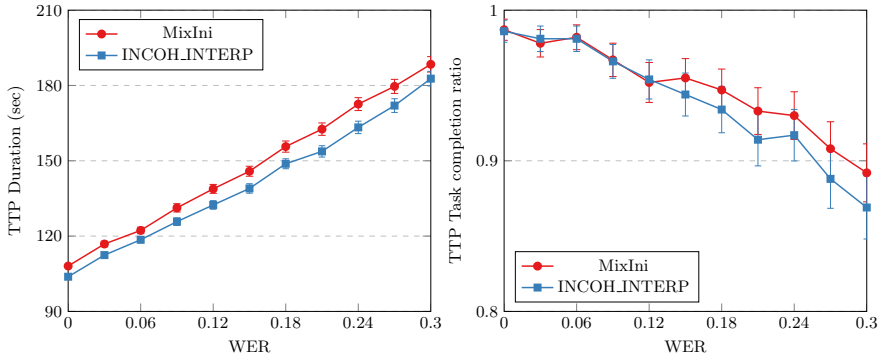


As can be expected, for low WER, the SysIni strategy is more tiresome and inefficient compared to UsrIni (Figure 1). However, for high noise levels, the ranks are reversed as it is more efficient to communicate the information chunk by chunk in an uncertain communication channel. MixIni has the advantages of both strategies as it performs like UsrIni for low WERs and better than both of them in noisy situations. Incremental behaviour has been introduced to both UsrIni and MixIni (in the case of SysIni, the user's utterances are much shorter, so it is less relevant to add incrementality). Just like mixed initiative, incrementality is also shown to be a solution to making UsrIni more robust to noise, with even better results. Finally, we show that it is possible to add incremental behaviour to MixIni which performs best.

In Figure 2, the performance of each turn-taking phenomenon is represented (and compared with MixIni as a baseline). FEEDBACK_RAW, BAGE_IN_RESP from the user's side and FAIL_RAW are the phenomena that have a real impact on the dialogue efficiency. This is due to the fact that the task is not very adapted to the two other phenomena. It has been slightly modified so that the probability of detecting an incoherence is higher. Figure 3 compares MixIni with INCOHERENCE_INTERP in the case of one scenario where the user tries to move an event five times before discovering a free slot. Therefore, most of her requests refer to an existing event and if the event title is modified because of ASR noise, an incoherence is detected. INCOHERENCE_INTERP reduces the dialogue duration. The task completion is not reduced significantly but consistently over the different WER levels (starting from WER=0.15). But in this task, MixIni already performs very well so the margin for improvement is small.

BAGE_IN_RESP from the system's side does not make the system more robust to noise as it does not handle errors. It is useful with users who tend to make unnecessarily long utterances. [5] shows that when the users are interrupted they tend to make more concise utterances focusing on the main information.

Fig. 3. INCOHERENCE_INTERP evaluated in a more adapted task (see Sec. 3).



4 Conclusion and future work

A simulated environment has been used to show that incremental dialogue is a way to combine the advantages of system and user initiative strategies. Moreover, we show that there is also room for improvement using incremental dialogue processing. A traditional dialogue system can be straightforwardly transformed into an incremental one by adding a Scheduler module between the client and the service [6]. And by embedding well selected incremental behaviours in this module, one can build more robust dialogue systems. In this work, hand-crafted rules have been used to implement these behaviours. In future work, we plan to use reinforcement learning to automatically learn optimal turn-taking management through the Scheduler decisions. Moreover, interactions with real users will be used for evaluation and training.

References

1. Allen, J., Ferguson, G., Stent, A.: An architecture for more realistic conversational systems. In: 6th international conference on Intelligent user interfaces (2001)
2. Asri, L.E., Laroche, R., Pietquin, O.: Reward function learning for dialogue management. In: STAIRS (2012)
3. DeVault, D., Sagae, K., Traum, D.: Incremental interpretation and prediction of utterance meaning for interactive dialogue. Dialogue and Discourse 2, 143–170 (2011)
4. Dohsaka, K., Shimazu, A.: A system architecture for spoken utterance production in collaborative dialogue. In: IJCAI (1997)
5. Ghigi, F., Eskenazi, M., Torres, M.I., Lee, S.: Incremental dialog processing in a task-oriented dialog. In: Fifteenth Annual Conference of the International Speech Communication Association (2014)
6. Khouzaimi, H., Laroche, R., Lefèvre, F.: An easy method to make dialogue systems incremental. In: Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL) (2014)

7. Schlangen, D., Skantze, G.: A general, abstract model of incremental dialogue processing. Dialogue and Discourse 2, 83–111 (2011)
8. Selfridge, E., Arizmendi, I., Heeman, P., Williams, J.: Continuously predicting and processing barge-in during a live spoken dialogue task. In: Proceedings of the SIGDIAL 2013 Conference (2013)
9. Skantze, G., Schlangen, D.: Incremental dialogue processing in a micro-domain. In: ACL (2009)
10. Tanenhaus, M.K., Spivey-Knowlton, M.J., Eberhard, K.M., Sedivy, J.C.: Integration of visual and linguistic information in spoken language comprehension. Science 268, 1632–1634 (1995)
11. Walker, M.A., Litman, D.J., Kamm, C.A., Abella, A.: Paradise: a framework for evaluating spoken dialogue agents. In: Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics (1997)